This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method in a computer system for servicing requests from one or more client computers, the method comprising:

receiving a request from a client computer;

a first thread processing the request by invoking a receive handler that creates a work item, wherein the first thread is part of a pool of generic threads;

a second thread performing a task specified in the work item by invoking a work handler, wherein the second thread is part of the pool of generic threads;

receiving a result of performing the task; ~~and~~

a third thread returning at least a portion of the result to the client computer by invoking a reply handler, wherein the third thread is part of the pool of generic threads;

determining a size of the result; and

when the size of the result is not larger than a cutoff size, storing the result in the partial results cache.

2. (Original) The method as claimed in claim 1, wherein receiving the request comprises:

receiving the request into an input/output port; and

placing a reference in a queue indicating that work is available for the first thread.

3. (Original) The method as claimed in claim 2, further comprising: invoking a receive thread manager when the work is available; and

the receive thread manager scheduling the first thread for execution on one of multiple processors.

4. (Original) The method as claimed in claim 3, wherein the receive thread manager schedules the first thread from a queue of available threads.

5. (Original) The method as claimed in claim 1, further comprising the first thread placing the work item on a work queue for execution by the second thread.

6. (Original) The method as claimed in claim 5, further comprising the first thread placing a reference in a completion port queue, indicating that work is available for the second thread.

7. (Original) The method as claimed in claim 1, further comprising the second thread creating and placing a second work item on a reply work queue when the results are received.

8. (Original) The method as claimed in claim 7, further comprising the second thread placing a reference in a completion port queue, indicating that work is available for the third thread.

9. (Original) The method as claimed in claim 1, further comprising:
receiving input data; and
the first thread storing the input data in a cache that is accessible to the second thread.

10. (Original) The method as claimed in claim 1, wherein the results include data, and further comprising storing the data in a cache that is accessible to the third thread.

11. (Cancelled)

12. (Currently Amended) The method as claimed in claim 1 ~~11~~, further comprising:
the third thread returning a portion of the result to client computer; and
if a second request is received for an additional portion of the result that is stored in the partial results cache, a fourth thread creating a second work item by invoking another receive handler, wherein the second work item is processed by a fifth thread, which invokes another reply handler.

13. (Original) The method as claimed in claim 1, further comprising:

the first thread, the second thread or the third thread indicating that the first thread, the second thread or the third thread has completed a work item; and

if a quantum has not expired for the first thread, the second thread or the third thread, then the first thread, the second thread or the third thread being given an additional work item to perform without relinquishing the central processing unit upon which the first thread, the second thread or the third thread was running.

14. (Cancelled)

15. (Cancelled)

16. (Cancelled)

17. (Cancelled)

18. (Cancelled)
19. (Cancelled)
20. (Cancelled)
21. (Cancelled)
22. (Cancelled)

23. (Cancelled)
24. (Cancelled)
25. (Cancelled)

26. (Currently Amended) A method in a computer system for servicing requests from multiple client computers, the method comprising:

determining that work is available after receiving a request from a client computer, wherein the request from the client computer is a request to perform a function having multiple states;

when work is available, a first work handler invoked by a first thread looking in a first work queue for a first work item corresponding to the work; ~~and~~

if the first work item is not found in the first work queue, the first work handler looking in a second work queue for the first work item, wherein the computer system includes multiple work queues, including the first work queue and the second work queue, each of the multiple work queues are associated with a priority level;

looking for the first work item first in a work queue associated with a highest priority level; and

if the first work item is not found in the work queue associated with the highest priority level, looking for the first work item in each of the multiple work queues in descending priority order until the first work item is found, wherein each state of the multi-state function is performed by a work handler invoked by a subsequent thread based on work items placed in the multiple work queues, and wherein the work items are placed in higher and higher priority work queues as execution of the multi-state function progresses through the multiple states.

27. (Original) The method as claimed in claim 26, further comprising:
receiving a request from a client computer to perform a task;
creating the first work item that specifies the task;
placing the first work item in the second queue; and
indicating that the work is available.

28. (Original) The method as claimed in claim 26, further comprising:
the first work handler performing a task specified in the first work item; and
issuing an asynchronous request for data.

29. (Original) The method as claimed in claim 28, further comprising:
receiving the data;

placing a second work item on the first work queue; and

indicating that additional work is available.

30. (Original) The method as claimed in claim 29, further comprising:

a second work handler invoked by a second thread looking in the first work queue for the second work item when the second work is available; and

performing a second task specified in the second work item.

31. (Cancelled)

32. (Cancelled)

33. (Currently Amended) A method in a computer system for servicing requests from multiple client computers, the method comprising:

receiving, from a client computer, a request to perform a first task;

evaluating the first task, by a first handler invoked by a first thread, to determine whether the first task includes complex or long-running logic; ~~and~~

if the first task includes complex or long-running logic, performing the first task by a second handler invoked by a second thread, wherein the first thread is a thread within a first group of threads having a first priority level, and the second thread is a thread within a second group of threads having a second priority level that is lower than the first priority level; and

receiving a request to perform a second task;

evaluating the second task to determine whether the second task includes complex or long-running logic; and

if the second task does not include complex or long-running logic and a processor is not available for performing the second task, preempting the second thread and performing the second task by a third thread in the first group of threads.

34. (Cancelled)

35. (Cancelled)

36. (Original) The method as claimed in claim 33, further comprising:

the second handler returning a result of the first task; and

using the result, performing a second task by a third handler invoked by a third thread of the first group of threads.

37. (Original) The method as claimed in claim 36, wherein the first task is specified in a first work item and the second task is specified in a second work item, the method further comprising:

the first handler obtaining the first work item from a first work queue; and

the third handler obtaining the second work item from a second work queue.

38. (Cancelled)

39. (Cancelled)

40. (Cancelled)

41. (Cancelled)

42. (Cancelled

43. (Cancelled)

44. (Cancelled

45. (Cancelled)

46. (Cancelled)

47. (Cancelled)

48. (Cancelled)

49. (Cancelled)

50. (Cancelled)

51. (Cancelled)

52. (Cancelled)

53. (Cancelled)

54. (Cancelled)

55. (Cancelled)

56. (Cancelled)

57. (Cancelled)

58. (Previously Presented) A computer system for servicing requests from one or more client computers, the computer system comprising:

a memory containing an application server which maintains a pool of threads, wherein each thread in the pool of threads is identical and can invoke at least one receive handler, at least one work handler, and at least one reply handler, and the application server further schedules threads in the pool of threads for execution on multiple processors available to the computer system; and

the multiple processors for simultaneously executing multiple threads within the pool of threads.

59. (Cancelled)

60. (Cancelled)

61. (Cancelled)

62. (Cancelled)

63. (Cancelled)

64. (Cancelled)

65. (Previously Presented) A computer-readable medium holding computer executable instructions, the computer-readable medium for performing a method in a computer system, the method comprising:

maintaining a pool of threads, wherein each thread in the pool of threads is identical and can invoke at least on receive handler, at least one work handler, and at least one reply handler, and the application server further schedules threads in the pool of threads for execution on multiple processors available to the computer system; and

simultaneously executing multiple threads within the pool of threads on multiple processors available to the computer system.

66. (Cancelled)

67. (Cancelled)

68. (Cancelled)

69. (Cancelled)

70. (Cancelled) .

71. (Cancelled)